発電システムの監視

SUNNY WEBBOX RPC

取扱説明書

リモートプロシージャコールについて
インターフェースと API の定義

SWebBoxRPC-BA-ja-15 | バージョン 1.5

JP

目次

1 はじめに 5
1.1 適用範囲 5
1.2 製品の表記について 5
1.3 アルファベット略語 5
1.4 参考文献 6
2 システム概要 7
2.1 Sunny WebBox RPC 7
2.2 Sunny WebBox と *Bluetooth* 搭載 Sunny WebBox の相違点 . 7
3 JavaScript Object Notation 8
3.1 例 8
4 プロシージャのルール 10
4.1 プロシージャコール（リクエスト） 10
4.2 リターン値（レスポンス） 11
4.2.1 JSON 11
4.3 クエリーの間隔 11
5 インターフェース 12
5.1 UDP ストリームを介した RPC 12
5.2 HTTP を介した RPC 12
6 オブジェクトの定義 13
6.1 デバイスオブジェクト 13
6.2 チャンネルオブジェクト 14
7 サービスプロシージャ 15
7.1 RPC_GET_PLANT_OVERVIEW 15
7.1.1 バージョン 1.0 15
7.2 RPC_GET_DEVICES 17
7.2.1 バージョン 1.0 17
7.3 RPC_GET_PROCESS_DATA_CHANNELS 19
7.3.1 バージョン 1.0 19

7.4 RPC_GET_PROCESS_DATA 21
7.4.1 バージョン 1.0 21
7.5 RPC_GET_PARAMETER_CHANNELS 24
7.5.1 バージョン 1.0 24
7.6 RPC_GET_PARAMETER 26
7.6.1 バージョン 1.0 26
7.7 RPC_SET_PARAMETER 31
7.7.1 バージョン 1.0 31

1 はじめに

1.1 適用範囲

本書は、Sunny WebBox および *Bluetooth*® Wireless Technology を搭載した Sunny WebBox の RPC インターフェースに適用されます。

1.2 製品の表記について

本書では、*Bluetooth*® Wireless Technology を搭載した Sunny WebBox を *Bluetooth* 搭載 Sunny WebBox と表記しています。

1.3 アルファベット略語

API	Application Programming Interface
HTTP	Hypertext Transfer Protocol
JSON	Javascript Object Notation
UDP	User Data Protocol
URL	Uniform Resource Locator
RAS	Remote Access Service
RFC	Request for Comment
RPC	Remote Procedure Call

1.4 参考文献

[1]
RFC 4627: The application/json Media Type for Javascript Object Notation (JSON)
http://www.ietf.org/rfc/rfc4627.txt?number=4627

[2]
JSON-RPC 1.1 Working Draft August 2006
http://json-rpc.org/wd/JSON-RPC-1-1-WD-20060807.html

[3]
Introducing JSON
http://www.json.org

[4]
The MD5 Message-Digest Algorithm
http://www.ietf.org/rfc/rfc1321.txt?number=1321

2 システム概要

2.1 Sunny WebBox RPC

Sunny WebBox および *Bluetooth* 搭載 Sunny WebBox は、太陽光発電システムの全データを継続的に記録するデータロガーです。記録されたデータから、設定された間隔で平均値が算出され、キャッシュに格納されます。このデータを定期的に Sunny Portal に送信し、分析やビジュアル化に使用することができます。

Sunny WebBox および *Bluetooth* 搭載 Sunny WebBox の RPC インターフェースを介し、太陽光発電システムのデータの中から選択したデータを、Remote Procedure Call（RPC）プロトコルを使って遠隔地のターミナルに送信することができます。

そのため、Sunny WebBox にはサービスプロシージャのプールがあり（13 ページの 6「オブジェクトの定義」を参照）、ネットワークまたは RAS 接続を介し、RPC プロトコルを使ってリモートターミナルからこのプールにアクセスが可能です。このデータ通信で使用されるフォーマットは JavaScript Object Notation（JSON）です（8 ページの 3「JavaScript Object Notation」を参照）。

2.2 Sunny WebBox と *Bluetooth* 搭載 Sunny WebBox の相違点

Sunny WebBox と *Bluetooth* 搭載 Sunny WebBox で異なる表示形態

RPC インターフェースを介して太陽光発電システムから送信されたデータの表示形態は、Sunny WebBox と *Bluetooth* 搭載 Sunny WebBox で異なります。

本書では、*Bluetooth* 搭載 Sunny WebBox ではなく、Sunny WebBox の応答例のみを使用しています。

Bluetooth 搭載 Sunny WebBox の測定値およびパラメータは、当社ホームページ（www.SMA-Solar.com）から技術資料「Measured Values and Parameters」にてご確認いただけます。

使用できるサービスプロシージャ	Sunny WebBox	*Bluetooth* 搭載 Sunny WebBox
RPC_GET_PLANT_OVERVIEW	+	+
RPC_GET_DEVICES	+	+
RPC_GET_PROCESS_DATA_CHANNELS	+	+
RPC_GET_PROCESS_DATA	+	+
RPC_GET_PARAMETER_CHANNELS	+	–
RPC_GET_PARAMETER	+	–
RPC_SET_PARAMETER	+	–

3 JavaScript Object Notation

JavaScript Object Notation（JSON）に関する詳細ついては、ウェブサイト http://www.json.org をご覧ください。

3.1 例

以下にデバイスリストの例を挙げます。ここでは「totalDevicesReturned」と「devices」の各値からなるオブジェクトを定義しています。

「totalDevicesReturned」は数字で、その値は「4」です。配列「devices」は 2 つのフィールドを有し、各フィールドにそれぞれ 1 つのデバイスオブジェクト（13 ページの 6.1「デバイスオブジェクト」を参照）が含まれ、各デバイスオブジェクトにさらにデバイスオブジェクトがネストされています。

```
RPC={
    "totalDevicesReturned":4,
    "devices":
    [
     {
      "key":"SCC250H9:1390148531",
      "name":"Sunny Central E1",
      "children":
      [
       {
        "key":"SCBFS016:8945",
        "name":"Sunny BFS E1",
        "children":null
       },
       {
        "key":"SMU8b004:2567",
        "name":"String Monitoring Unit E1",
        "children":null
       }
      ]
     },
     {
      "key":"SCC250H9:1390148538",
      "name":"Sunny Central E2",
      "children":
```

```
    [
     {
      "key":"SCBFS016:8956",
      "name":"Sunny BFS E2",
      "children":null
     },
     {
      "key":"SMU8b004:2534",
      "name":"String Monitoring Unit E2",
      "children":null
     }
    ]
   }
  ]
}
```

4 プロシージャのルール

使用されるすべての識別子で大文字小文字が区別されます。つまり、「Power」と「power」では異なる 2 つのオブジェクトを指定することになります。文字はすべて Unicode の UTF-8 形式を使って送信されます。

4.1 プロシージャコール（リクエスト）

各リクエストはシリアル化された 1 つの JSON オブジェクトから成り、このオブジェクトには必ず以下のメンバーが含まれていなければなりません。

- **version** — 基本となる RPC バージョンを定義する文字列。
- **proc** — コールするプロシージャを含む文字列。
- **id** — リクエストに対するレスポンスの指定に使用される任意の文字列（最大 16 文字）。
- **format** — プロシージャの結果を送信するための通信形式を定義する文字列（11 ページの 4.2「リターン値（レスポンス）」を参照）。
- **passwd** — 使用したいアクセスレベル（ユーザー、施工者）に指定されたパスワードのハッシュ値を含む文字列。ハッシュ値は MD5 アルゴリズムを使用して計算されます（[4] を参照）。パスワードの指定は Sunny WebBox の安全用設定値を介して行います。このオブジェクトが送信されないと、アクセスレベルは自動的に「ユーザー」となります。
- **params** — 引数としてプロシージャに送信されるエレメントを有するオブジェクト。各パラメータは、それぞれ指定された JSON オブジェクトの形に対応している必要があります。そのため、順番は任意です。パラメータの数は、サービスプロシージャの種類によって決まります（第 7 章を参照）。リクエストされたプロシージャでパラメータが必要ない場合は省略されます。

4.2 リターン値（レスポンス）

レスポンスのデータ通信形式は、リクエストで送信された文字列により定義されます。

現在、以下の形式が使用できます。

- JSON

4.2.1 JSON

リクエストに対するレスポンスはシリアル化された 1 つの JSON オブジェクトから成り、これには必ず以下のオブジェクトが含まれていなければなりません。

- **version** — 基本となる RPC バージョンを定義する文字列。
- **proc** — コールされたプロシージャを含む文字列。
- **id** — リクエストおよびレスポンスの指定に使用される文字列。対応するリクエストの ID が含まれています。
- **result** — シリアル化された JSON オブジェクトとして得られる、実行されたプロシージャの結果。エラーが発生してプロシージャが正常に実行できなかった場合には、代わりに「error」オブジェクトが送信されます。
- **error** — 発生したエラーを記述する文字列を含むオブジェクト。プロシージャが実行されてエラーが発生しなかった場合、このオブジェクトは送信されません。

4.3 クエリーの間隔

クエリーとクエリーの間には最低 30 秒の間隔をあける必要があります。

5 インターフェース

Sunny WebBox では 2 つのアクセス方法が使用できます。各方法で、実装に必要となる手間およびランタイムリソースの消費量が異なります。

5.1 UDP ストリームを介した RPC

プロシージャコールは、UDP プロトコルの実際のデータ部分を使用して Sunny WebBox のポート 34268 に送信されます。レスポンスもポート 34268 に返されます。

UDP を使ったトランスポートでは、クライアント側での実装にそれほど手間がかからず、またランタイムリソースも節約できます。複数のローカルネットワークをまたぐ通信においては、適切なファイアウォールを使用してポートを開く必要があります。

5.2 HTTP を介した RPC

Sunny WebBox で設定された Web サーバーポートへの TCP/IP 接続を介し、Hypertext Transfer Protocol（HTTP）を使用してデータ通信を行います。

デフォルトではポート 80 が設定されています。

すべてのリクエストで以下の URL が使用されます : http://IP アドレス /rpc

個々のケースで使用する IP アドレスは、Sunny WebBox で実際に設定されている IP アドレスになります。デフォルトでは 192.168.0.168 に設定されています。

そのため、デフォルト設定で使用する URL は以下のようになります : http://192.168.0.168/rpc

リクエストは、HTTP のボディにある HTTP POST を使用し、4.1 に規定されたルールに従い、シリアル化された JSON オブジェクトとして送信されます。

クライアント側で実装にかかる手間も、必要となるリソースも比較的大きくなります。原則として通信は標準ポート 80 を介して行われるため、ファイアウォールを有効にするための変更が不要です。

6 オブジェクトの定義

本章では、JSON 構文を使ったオブジェクトの中で、使用頻度の高いオブジェクトの構造を定義します。記述の中では、オブジェクトメンバーの値が使用データのタイプを定義します。すべての定義は、同様に、他のあらゆるデータ通信形式にも適用されます。

6.1 デバイスオブジェクト

太陽光発電システム内のデバイスを記述します（Sunny Boy、Sunny Sensor Box など）。

RPC={

"key": "string",

"name": "string" または null (任意),

"channels": [array] または null (任意),

"children": [array] または null (任意)

}

key: 一意のデバイスキー（「SB21TL06:2000106925」など）。

name: ユーザーが定義できるデバイス名（「パワコン左」など）。このエレメントの定義は任意です。このエレメントが使用されているのに名前が定義されていない場合は、「null」が入力されます。

channels: デバイスのチャンネルオブジェクトの配列。このエレメントの定義は任意です。このエレメントが使用されているのにチャンネルが存在しない場合は、「null」が入力されます。

children: サブデバイスを含むオブジェクトの配列。このエレメントの定義は任意です。このエレメントが使用されているのにサブデバイスが存在しない場合は、「null」が入力されます。

6.2 チャンネルオブジェクト

デバイスのプロセスデータまたはパラメータチャンネルを記述します。

```
RPC={
      "meta": "string",
      "name": "string" ( 任意 ),
      "value": "string" ( 任意 ),
      "unit": "string" ( 任意 ),
      "min": "string" ( 任意 ),
      "max": "string" ( 任意 ),
      "options": [array] ( 任意 )
    }
```

meta: チャンネルを一意に定義するメタネーム（「ExtSollrr」など）。

name: 翻訳された表示名（「External irradiation」など）。このエレメントの定義は任意です。

value: チャンネルの値（「843」など）。このエレメントの定義は任意です。

unit: チャンネルの単位（「W/m^2」など）。単位のないチャンネルでは、空白文字列を入力する必要があります。このエレメントの定義は任意です。

min: チャンネルで可能な最小値。このエレメントの定義は任意です。

max: チャンネルで可能な最大値。このエレメントの定義は任意です。

options: パラメータチャンネルで可能な値のリスト。このエレメントの定義は任意です。

7 サービスプロシージャ

本章では、使用できるサービスプロシージャの構造について説明しています。

各プロシージャで、その役割を簡単に記載しています。これに、リクエストの構造が続きます。変数エレメントは、大文字のプレースホルダで表されています。本説明書では、VERSION、FORMAT および ID の各プレースホルダをプロシージャごとに説明することはしていません。これらプレースホルダの意味については、すでに 4 章に記載があり、必ずしもすべてのプロシージャでその違いを述べる必要がないためです。

7.1 RPC_GET_PLANT_OVERVIEW

7.1.1 バージョン 1.0

太陽光発電システムに関する以下データを含むオブジェクトを返します。

- POWER
- DAILY-YIELD
- TOTAL-YIELD
- STATUS
- ERROR

構造：

```
RPC={
    "version": "1.0",
    "proc": "GetPlantOverview",
    "id": "ID",
    "format": "FORMAT"
  }
```

リクエスト例：

```
RPC={
    "version": "1.0",
    "proc": "GetPlantOverview",
    "id": "1",
    "format": "JSON"
  }
```

レスポンス例：

```
RPC={
    "version": "1.0",
    "proc": "GetPlantOverview",
    "id": "1",
    "result":
    {
     "overview":
     [
      {
       "meta": "GriPwr",
       "name": "instantaneous power",
       "value": "4250",
       "unit": "W"
      },
      {
       "meta": "GriEgyTdy",
       "name": "daily energy",
       "value": "45.23",
       "unit": "kWh"
      },
      {
       "meta": "GriEgyTot",
       "name": "total energy",
       "value": "7821",
       "unit": "kWh"
      },
      {
       "meta": "OpStt",
       "name": "Status",
       "value": "MPP",
       "unit": null
      },
      {
       "meta": "Msg",
```

```
        "name": "Fault",
        "value": null,
        "unit": null
       }
      ]
     }
    }
```


以下のデータが送信されました。

Power = 4,250 W,
E-Today = 45.23 kWh,
E-Total = 7,821 kWh,
Status = MPP,
エラー無し

7.2 RPC_GET_DEVICES

7.2.1 バージョン 1.0

太陽光発電システムで検出されたすべてのデバイスを含む階層的リストを返します。

構造：


```
RPC={
    "version": "1.0",
    "proc": "GetDevices",
    "id": "ID",
    "format": "FORMAT"
   }
```


リクエスト例：


```
RPC={
    "version": "1.0",
    "proc":  "GetDevices",
    "id": "1",
    "format": "JSON"
   }
```

レスポンス例：


```
RPC={
    "version": "1.0",
    "proc": "GetDevices",
    "id": "1",
    "result":
    {
     "totalDevicesReturned": 6,
     "devices":
     [
      {
       "key":"SCC250H9: 1390148531",
       "name":" Sunny Central E1",
       "children":
       [
        {
         "key": "SCBFS016:8945",
         "name": "Sunny BFS E1",
         "children": null
        },
        {
         "key": "SMU8b004:2567",
         "name": "String Monitoring Unit E1",
         "children": null
        }
       ]
      },
      {
       "key": "SCC250H9:1390148538",
       "name": "Sunny Central E2",
       "children":
       [
        {
         "key": "SCBFS016:8956",
         "name": "Sunny BFS E2",
```

```
        "children": null
       },
       {
        "key": "SMU8b004:2534",
        "name": "String Monitoring Unit E2",
        "children": null
       }
      ]
     }
    ]
   }
  }
```


7.3 RPC_GET_PROCESS_DATA_CHANNELS

7.3.1 バージョン 1.0

特定のデバイスタイプで、使用できるプロセスデータチャンネルのメタネームを含むリストを返します。

構造：


```
RPC={
    "version": "1.0",
    "proc": "GetProcessDataChannels",
    "id": "ID",
    "format": "FORMAT",
    "params":
    {
     DEVICE_KEY
    }
   }
```


DEVICE_KEY: プロセスデータチャンネルを返すべきデバイスタイプに対応するデバイスキー。

リクエスト例：

```
RPC={
      "version": "1.0",
      "proc": "GetProDataChannels",
      "id": "1",
      "format": "JSON",
      "params":
      {
        "device": "WR715-19:263415747"
      }
    }
```

レスポンス例：

```
RPC={
      "version": "1.0",
      "proc": "GetProcessDataChannels",
      "id": "1",
      "result":
      {
        "WR715-19:263415747":
        [
          "Upv-Soll",
          "h-Total",
          "Zac",
          "Status",
          "E-Total",
          "Upv-Ist",
          "Riso",
          "Uac",
          "Pac",
          "Fehler-Cnt",
          "Ipv",
          "Netz-Ein",
          "Serial number",
          "Fac"
          "Fault"
```

```
      "Iac-Ist"
     ]
    }
   }
```

7.4 RPC_GET_PROCESS_DATA

7.4.1 バージョン 1.0

リクエスト 1 件につき、最大 5 つのデバイスのプロセスデータを返します。

構造：

```
RPC={
    "version": "1.0",
    "proc": "GetProcessData",
    "id": "ID",
    "format": "FORMAT",
    "params":
    {
     "DEVICES":
     [
      {
       "key": DEVICE_KEY,
       "channels": [CHANNELS]
      }
     ]
    }
   }
```

プロセスデータを返すべきデバイスキーのリストは、パラメータとして送信する必要があります。各デバイスに必要なプロセスデータを選択して送信することができます。選択を省略すると、すべてのプロセスデータが送信されます。

DEVICES: プロセスデータを返すべきデバイスのデバイスキー、および任意で CHANNELS を含むオブジェクトからなる配列。

DEVICE_KEY: 対応するデバイスキー（13 ページの 6.1「デバイスオブジェクト」を参照）。

CHANNELS: 必要なプロセスデータのメタネームを含む配列。使用できるメタネームは、コマンド RPC_GET_PROCESS_DATA_CHANNELS を使って識別することができます。

リクエスト例：

```
RPC={
      "version": "1.0",
      "proc": "GetProcessData",
      "id": "1",
      "format": "JSON",
      "params":
      {
       "devices":
       [
        {
         "key": "WR715-19:263415747",
         "channels": null
        },
        {
                    "key": "WR715-19:263415748",
         "channels":
         [
          "Pac"
         ]
        }
       ]
      }
     }
```

レスポンス例：

```
RPC={
      "version": "1.0",
      "proc": "GetProcessData",
      "id": "1",
      "result":
      {
       "devices":
       [
        {
```

```
 "key": "WR715-19:263415747",
 "channels":
 [
  {
   "meta": "E-Total",
   "name": null,
   "value": "1160.987",
   "unit": "kWh"
  },
  {
   "meta": "Fac",
   "name": null,
   "value": "49.98",
   "unit": "Hz"
  },
  {
   "meta": "Zac",
   "name": null,
   "value": "1.346",
   "unit": "Ohm"
  }
 ]
},
{
        "key": "WR715-19:263415748",
 "channels":
 [
  {
   "meta": "Pac",
   "name": null,
   "value": "630",
   "unit": "W"
  }
 ]
}
```

```
    ]
   }
  }
```

7.5 RPC_GET_PARAMETER_CHANNELS

7.5.1 バージョン 1.0

対応するアクセスレベルに応じ、特定のデバイスタイプで、使用できるパラメータチャンネルのメタネームを含むリストを返します。アクセスレベルは、リクエストのヘッダーで対応するパスワードの MD5 ハッシュ値を送信することにより特定されます。

構造：

```
RPC={
    "version": "1.0",
    "proc": "GetParameterChannels",
    "id": "ID",
    "format": "FORMAT",
    "passwd" : "PASSWORD",
    "params":
    {
     "key": DEVICE_KEY
    }
   }
```

PASSWORD: 使用したいアクセスレベルに指定されたパスワードの、MD5 でコード化されたハッシュ値。

DEVICE_KEY: パラメータチャンネルを返すべきデバイスタイプに対応するデバイスキー。

リクエスト例：

```
RPC={
    "version": "1.0",
    "proc": "GetParameterChannels",
    "id": "1",
    "format": "JSON",
    "passwd": "a289fa4252ed5af8e3e9f9bee545c172",
    "params":
```

```
    {
     "device": "WR715-19:263415747"
    }
   }
```

レスポンス例：

```
RPC={
    "version": "1.0",
    "proc": "GetParameterChannels",
    "id": "1",
    "result":
    {
     "WR715-19:263415747":
     [
      "Plimit",
      "SMA-Grid-Guard",
      "SMA-SN",
      "Operating mode",
      "Control",
      "Ripple-Ctl-Frq",
      "PowerBalancer",
      "Usoll-Konst",
      "Upv-Start",
      "Default",
      "T-Start",
      "Ripple-Ctl-Lev",
      "Storage",
      "Ripple-Ctl-Rcvr",
      "Software-SRR",
      "T-Stop",
      "Software-BFR",
      "Hardware-BFS"
     ]
    }
   }
```

7.6 RPC_GET_PARAMETER

7.6.1 バージョン 1.0

アクセスレベルに応じ、最大 5 つのデバイスのパラメータ値を返します。アクセスレベルは、リクエストのヘッダーで対応するパスワードの MD5 ハッシュ値を送信することにより特定されます。

構造：

```
RPC={
      "version": "1.0",
      "proc": "GetParameter",
      "id": "ID",
      "format": "FORMAT",
      "passwd": "PASSWORT",
      "params":
      {
       "DEVICES":
       [
        {
         "key": DEVICE_KEY,
         "channels": [CHANNELS]
        }
       ]
      }
    }
```

パラメータを返すべきデバイスオブジェクトのリストは、パラメータとして送信する必要があります。各デバイスで必要なパラメータを選択して送信することができます。選択を省略すると、すべてのパラメータが送信されます。

PASSWORD: 使用したいアクセスレベルに指定されたパスワードの、MD5 でコード化されたハッシュ値。

DEVICES: パラメータを返すべきデバイスオブジェクトと、任意で、選択した特定のチャンネルを含む配列。

DEVICE_KEY: 対応するデバイスキー（13 ページの 6.1「デバイスオブジェクト」を参照）。

CHANNELS: 必要なプロセスデータのメタネームを含む配列。使用できるメタネームは、コマンド RPC_GET_PROCESS_DATA_CHANNELS を使って識別することができます。

リクエスト例：

```
RPC={
      "version": "1.0",
      "proc": "GetParameter",
      "id": "1",
      "format": "JSON",
      "passwd": "a289fa4252ed5af8e3e9f9bee545c172",
      "params":
      {
       "devices":
       [
        {
         "key": "WR715-19:263415747"
        }
       ]
      }
    }
```

レスポンス例：

```
RPC={
      "version": "1.0",
      "id": "1",
      "format": "JSON",
      "proc": "GetParameter",
      "result":
      {
       "devices":
       [
        {
         "key": "WR21TL06:2000101000"
         "channels":
         [
          {
           "min": "0",
           "max": "7",
           "meta": "Operating mode",
```

```
 "options":
 [
  "Stop",
  "Konstantspg.",
  "Mpp-Betrieb",
  "Res1",
  "Res2",
  "Res3",
  "Res4",
  "Res5"
 ],
 "value": "Mpp-Betrieb",
 "name": "Operating mode",
 "unit": ""
},
{
 "min": "2150",
 "max": "2150",
 "meta": "Plimit",
 "value": "2150",
 "name": "Plimit",
 "unit": "W"
},
{
 "min": "0",
 "max": "4294900000",
 "meta": "SMA-SN",
 "value": "2000101000",
 "name": "SMA-SN",
 "unit": ""
},
{
 "min": "125",
 "max": "600",
 "meta": "Upv-Start",
```

```
  "value": "150",
  "name": "Upv-Start",
  "unit": "V"
 },
 {
  "min": "1",
  "max": "300",
  "meta": "T-Stop",
  "value": "4",
  "name": "T-Stop",
  "unit": "s"
 },
 {
  "min": "125",
  "max": "600",
  "meta": "Usoll-Konst",
  "value": "600",
  "name": "Usoll-Konst",
  "unit": "V"
 },
 {
  "min": "5",
  "max": "300",
  "meta": "T-Start",
  "value": "10",
  "name": "T-Start",
  "unit": "s"
 },
 {
  "min": "0",
  "max": "100",
  "meta": "Software-SRR",
  "value": "2",
  "name": "Software-SRR",
  "unit": "Version"
```

```
},
{
 "min": "0.005",
 "max": "4",
 "meta": "dFac-Max",
 "value": "0",
 "name": "dFac-Max",
 "unit": "Hz/s"
},
{
 "min": "0",
 "max": "7",
 "meta": "Storage",
 "options":
 [
  "permanent",
  "volatile",
  "Res1",
  "Res2",
  "Res3",
  "Res4",
  "Res5",
  "Res6"
 ],
 "value": "permanent",
 "name": "Storage",
 "unit": ""
},
{
 "min": "0",
 "max": "100",
 "meta": "Software-BFR",
 "value": "2",
 "name": "Software-BFR",
 "unit": "Version"
```

```
        },
        {
          "min": "0",
          "max": "100",
          "meta": "Hardware-BFS",
          "value": "1",
          "name": "Hardware-BFS",
          "unit": "Version"
        }
       ]
      }
     ]
    }
  }
```

7.7 RPC_SET_PARAMETER

7.7.1 バージョン 1.0

最大 5 つのデバイスでパラメータ値をセットし、リクエストで送信されたデバイスリストを、対応する現在のパラメータ値と一緒に返します。パラメータ値がすべて正しくセットされたかどうかのチェックは、使用しているアプリケーションで実行する必要があります。

構造：

```
RPC={
    "version": "1.0",
    "proc": "SetParameter",
    "id": "ID",
    "format": "FORMAT",
    "passwd": "PASSWORT",
    "params":
    {
     "DEVICES":
     [
      {
       key,
```

```
        "channels": [CHANNELS]
      }
     ]
    }
   }
```

パラメータを変更すべきデバイスオブジェクトのリストは、パラメータとして送信する必要があります。各デバイスオブジェクトには、セットすべきパラメータのリストが含まれます。

パラメータの設定は同時に実行されます。そのため、応答時間は、セットすべきパラメータの数に応じて変化します。以下に挙げた例では、応答時間が約 10 秒です。

PASSWORD: 使用したいアクセスレベルに指定されたパスワードの、MD5 でコード化されたハッシュ値。

DEVICES: デバイスのデバイスキーを含むオブジェクトの配列と、パラメータをセットする CHANNELS を含む配列。

DEVICE_KEY: 対応するデバイスキー（13 ページの 6.1「デバイスオブジェクト」を参照）。

CHANNELS: 対応するデバイスでセットすべきチャンネルオブジェクトを含む配列。使用できるパラメータチャンネルのリストは、コマンド RPC_GET_PARAMETER_CHANNELS を使って取得できます。

リクエスト例：

```
RPC={
    "version": "1.0",
    "proc": "SetParameter",
    "id": "1",
    "format": "JSON",
    "passwd": "a289fa4252ed5af8e3e9f9bee545c172",
    "params":
    {
     "devices":
     [
      {
       "key": "WR21TL06:2000101000",
       "channels":
       [
```

```
          {
            "meta": "Operating mode",
            "value": "Mpp-Betrieb"
          }
        ]
      },
      {
        "key": "WR21TL06:2000101001",
        "channels":
        [
          {
            "meta": "Operating mode",
            "value": "Stop"
          }
        ]
      }
    ]
  }
}
```


レスポンス例：


```
RPC={
    "version": "1.0",
    "id": "1",
    "format": "JSON",
    "proc": "SetParameter",
    "result":
    {
      "devices":
      [
        {
          "key": "WR21TL06:2000101000",
          "channels":
          [
            {
              "min": "0",
```

```
      "max": "7",
      "meta": "Operating mode",
      "options":
      [
       "Stop",
       "Konstantspg.",
       "Mpp-Betrieb",
       "Res1",
       "Res2",
       "Res3",
       "Res4",
       "Res5"
      ],
      "value": "Mpp-Betrieb",
      "name": "Operating mode",
      "unit": ""
     }
    ]
   },
   {
    "key": "WR21TL06:2000101001",
    "channels":
    [
     {
      "min": "0",
      "max": "7",
      "meta": "Operating mode",
      "options":
      [
       "Stop",
       "Konstantspg.",
       "Mpp-Betrieb",
       "Res1",
       "Res2",
       "Res3",
```

```
            "Res4",
            "Res5"
          ],
          "value": "Stop",
          "name": "Operating mode",
          "unit": ""
        }
      ]
    }
  ]
 }
}
```

法的制約

本書に記載されている情報は、SMA Solar Technology AG の所有物です。その内容を公開する場合は一部または全部かの別を問わず、SMA Solar Technology AG の書面による許可が必要です。製品の評価または正規の導入を目的として、社内向けに文書をコピーすることは認められており、許可を必要としません。

適合宣言書

SMA Solar Technology AG は、記述された装置が、1999/5/EC 指令で定める必須要求事項および関連規定に適合していることを、ここに宣言します。適合宣言書の全文は、www.SMA-Solar.com の製造者保証にてご覧いただけます。

商標

すべての商標は、当該表示にその旨が記載されていない場合でも適用されます。商標の指定がないことによって、製品またはブランドが登録商標ではないことを意味するものではありません。

Bluetooth® およびそのロゴは、Bluetooth SIG, Inc. の登録商標であり、SMA Solar Technology AG は本商標のいかなる使用も許可されています。

QR Code® は、株式会社デンソーウェーブの登録商標です。

SMA Solar Technology AG
Sonnenallee 1
34266 Niestetal
ドイツ

Tel. +49 561 9522-0
Fax +49 561 9522-100
www.SMA.de
E-Mail: info@SMA.de

© 2004 - 2013 SMA Solar Technology AG. All rights reserved

SMA Solar Technology

www.SMA-Solar.com